Journal of International Information Management

Volume 11 | Issue 2 Article 7

2002

A Simple Set Model of Requirements Representation

Jia-Lang Seng
National Chengchi University

A Simple Set Model of Requirements Representation

Jia-Lang Seng
National Chengchi University

ABSTRACT

A successful information systems development (ISD) depends on a complete, correct, and consistent set of requirements specification. In this paper, we present a set model of requirements representation to present functional and non-functional requirements in a systematic and schematic manner. Because the nature of the set operation is mathematical and methodological, requirements representation becomes scholastic and structured. In this paper, we first describe an incremental and iterative process model of requirements analysis where a spiral of requirements acquisition, articulation, and analysis is defined. We then depict a set-oriented data and process model to abstract the analyzed requirements into a hierarchical structure. We further delineate the set model to represent the abstracted functional and non-functional requirements. The main advantage of the simple set model is that it provides a set of tractable requirements specification that allows developers to test if the specification is complete, correct, and consistent. A simple example is prepared as an illustration of this approach to show the meaning, values, and feasibility.

INTRODUCTION

Requirements analysis (RA) is the first and the most critical phase in the information systems development (ISD). ISD represents information systems analysis and design that a complex organizational process whereby computer-based information systems are developed and maintained. Past and present study indicates that errors, mistakes, and changes occur in the early phase of requirements analysis consume the most portion of the information systems resources. The rework and redesign of information systems means unacceptable and unaffordable delay and waste to management. Requirements analysis is known as a difficult area of technical and behavioral challenges. The technical challenge comes from the nature that requirements are textual and bulky; requirements are uncertain and unknown; and requirements are constantly changing. The behavioral challenge comes from the difficulty that users cannot completely, correctly, and consistently articulate their requirements, nor can analysts acquire requirements in a complete, correct, and consistent manner.

The challenge represents a fundamental and structural problem faced by the requirements analysis for many years. It indicates requirements can no longer be determined at one single

phase in the conventional Waterfall systems development life cycle and requirements have to be determined incrementally and iteratively over the spiral of systems development life cycle (SDLC) with rapid prototyping. Furthermore, requirements analysis differs from design and should be handled in a distinct manner. Requirements analysis is a discovery process and a process of problem definition. Design is an alternative selection process and a process of problem resolution. A structured and systematic approach to requirements analysis is important. A tractable set of requirements specification is vital to produce a specification of requirements that is complete, correct, and consistent.

In this paper, we first describe an incremental and iterative process model of requirements analysis. Three main steps are defined in the process model to conduct the requirements acquisition, articulation, and analysis tasks. Following the process model, we delineate a set-oriented data and process model that is developed to abstract the analyzed requirements. These requirements are abstracted into functional and non-functional requirements components. We then depict a new approach to organize them into a hierarchical structure to prepare a base for the requirements representation. We apply the set logic and operation, and create a set model to represent the requirements components. A simple example of campus recruiting requirements analysis is prepared as an illustration of this approach to show the meaning, values, and feasibility.

This paper is organized into eight sections. Section one introduces the background and motivation of this research. Section two reviews the relevant requirements analysis methods. Section three describes an incremental and iterative process model of requirements analysis. Section four delineates a simple set model to represent the functional requirements in a hierarchy. Section five depicts a simple set model to represent non-functional requirements with symbolic logic and algorithm. Section six describes a simple example of illustration of this approach. Section seven discusses the contributions and limitations of the method. Section eight concludes the paper with a brief summary and the future research directions.

LITERATURE REVIEW

The past and present research on requirements analysis can be summarized into the following six question-answer categories.

(1) What to get? - Contents
What requirements to get or what are the contents of a requirements specification?
(2) How to get? – Methods
How to get requirements or what is the method to acquire requirements?
(3) How to represent? – Formulation
How to represent requirements or what is the method to formulate requirements?
(4) How to organize? – Repository
How to organize requirements or what is the method to store and access requirements?
(5) How to test? – Validation and Verification

How to test requirements and what is the method to test the completeness, correctness, and consistency of requirements?
(6) How to support the other information systems development phases from requirements analysis?

What to Get?

The first category of requirements research centers on the content and containment of requirements specification. The taxonomy analysis is described in [Emam and Madhavji 1994] that discusses the scope and scale of requirements results. A distinction is made between information systems development analysis activity and design activity. The former is a "what" activity to discover the characteristics of an information system. The later is a "how" activity to decide the alternatives of an information system. Classification study is the focus. It uses the domain study and feasibility study to carry out the mission as discussed in [Boehm, Bose, Horowitz, and Lee 1994].

How to Get?

The early research on "how to get" focuses on the development of general technique of data collection such as interviewing, brainstorming, survey, observation, meeting, and projection described in [Davis 1982] and [Yeh 1982]. Later research work centers on information acquisition techniques. These techniques work with specific analysis and design methods. They are classified as process-oriented, data-oriented, control-oriented, and object-oriented techniques. The well-known process-oriented approach is the data flow diagram (DFD). The representative data-oriented approach is the entity-relationship diagram (ERD). The typical control-oriented approach is the Jackson systems development method (JSD). And, the standard object-oriented approach is the unified modeling language (UML). Other relevant studies on requirements elicitation have a general focus on scenario and condition based information elicitation as described in [Gough, Fodemski, Higgins, and Ray 1995] [Rawsthorne 1996] [Nissen, Jeusfeld, Marke, Zemanek, and Huber 1996] and [Hall, Jackson, Lane, Nuseibeh, and Rapanotti 2002].

In the field of formal methods, use case and business event has been used to analyze requirements as described in [Farbey and Finkelstein 2001] [Nuseibeh, Easterbrook, and Russo 2000] and [Nuseibeh and Easterbrook 2000]. Volere process model focuses on understanding and modeling business problems instead of moving right away to the nuts and bolts of implementation, readers can benefit from the concepts of "trawling" (a requirements-gathering process), quality gateways, and the use of templates to help simplify the process as described in [Robertson and Robertson 2000]. Viewpoint is applied in process model to capture requirements as described in [Finkelstein, Gabbay, Hunter, Kramer, and Nuseibeh 1994] [Nuseibeh, Kramer, and Finkelstein 1994] and the frame approach to requirements description in [Hall, Jackson, Lane, Nuseibeh, and Rapanotti 2002].

How to Represent?

Requirements representation is to formulate the acquired and analyzed requirements in a systematic and schematic manner. Requirements specification language or requirements repre-

sentation language is the main approach adopted to formulate the articulated requirements. Well-known requirements specification languages such as the early work of PSL (program statement language) and SDL (systems definition language) described in [Yau and Tsai 1986], an executable specification language called PAISLey described in [Zave 1991], and a method-independent SRS language (software requirements specification) described in [Davis 1990] are well represented and domain specific. Knowledge-based requirements representation described in [Hudlicka 1996] is a rule-based approach to model the unstructured and dynamic aspect of functional requirements. [Russo, Miller, Nuseibeh, and Kramer 2002] created an event-based requirements specification method to model the functional requirements based on deductive behavior of events.

How to Organize?

How to organize requirements is an issue of how to store and access requirements. Requirements repository is a common approach adopted to tackle the issue. Intelligent repository such as the knowledge-based requirements assistant (KBRA) described in [Czuchry and Harry 1988] and the knowledge apprentice (KA) described in [Reubenstein and Waters 1991] are representative of relevant and reusable requirements repository. Rapid prototyping is a quick interface design to present the modeled requirements [Ramesh and Luqi 1993]. Intent specification described in [Leveson 1998] is a new approach to organize the elicited and extracted requirements into a reusable form of repository. Reasoning in inconsistency is another approach representing requirements [Menzies, Easterbrook, Nuseibeh, and Waugh 1999]. [Nentwich, Capra, Emmerich, and Finkelstein 2002] describes an xlink-based technique to linkbase the requirements components as a viable construct of relationship in the distributed web application requirements analysis. [Sutcliffe 2001] addresses a wider scale of software requirements organization issue to balance between social behavior understanding and technical structure abstraction that can be tackled from multiple views and scenarios analysis then build multi-facet artifact to allow recurring merge and match.

How to Test?

Requirements validation is to validate and verify the collected requirements and test if they are complete, correct, and consistent. Completeness means no requirements are left out. Input, process, and output model is used to test the requirements completeness. Correctness suggests no requirements differ from the original sources. Consistency indicates no requirements that mean one thing at a place mean another in another place. Validation is a difficult area. Ad hoc approaches such as manual review, inspection, and walk through are usually adopted. Little literature is found to elaborate on how a complete and comprehensive validation and verification method is concluded. [Bastani, DeMarco, and Pasquini 1993] describes a fuzzy set approach to tackle the correctness issue in software quality. However, no representation method is suggested to provide an integrated solution. [Schneider, Easterbrook, Callahan, and Holzmann 1998] is a model checking approach to examine errors and mistakes for a fault tolerance system that however is a special industrial case study? Hence, an integral, general,

and mathematical method of set model to produce tractable requirements is critical.

Non-Functional Requirements Analysis

Non-functional requirements analysis has not been the focus of information systems development research. One reason is due to a mis-conception that non-functional requirements are usually non-quantifiable. As we summarize in the Appendix A, there are six categories of non-functional requirements and constraints. Three out of the six categories are quantifiable. For example, part of the design constraints category; timing, space, reliability, availability, accuracy, cost benefit analysis, physical constraints in the performance requirements category; and the economic constraints category are mostly quantifiable. Another reason is due to a mis-conception that functional requirements are considered the core of requirements analysis. The myth continues with a belief that if functional requirements analysis is completed, the non-functional requirements analysis can be assumed complete. Examining the appendix, we find that non-functional requirements are partially dependent on functional requirements such as the performance requirements and the implementation requirements, and partly independent of functional requirements such as the design constraints, the development requirements, and the managerial requirements. In fact, design, development, and management requirements represent non-functional requirements that drive and direct the overall requirements analysis process. Hence, non-functional requirements analysis is not a part of functional requirements analysis nor replaced by functional requirements analysis. However, in practice, non-functional requirements analysis is often less focused and fulfilled. The complexity and diversity of non-functional requirements is another reason why this subject is less treated [Davis 1994].

[Nixon 1993] gives a taxonomy of performance requirements but without suggesting a methodological approach. [Chung, Nixon, and Yu 1995] describes the relationship between non-functional requirements and change management in order to suggest a possible framework of non-functional requirements determination. [Robertson and Robertson 1999] details practices in capturing non-functional requirements along with functional requirements.

AN INCREMENTAL AND ITERATIVE PROCESS MODEL OF REQUIREMENTS ANALYSIS

Information systems development (ISD) is a process of analysis, design, and implementation. An information systems development process model defines the order, duration, and transition of various development activities. Alternative information systems development process models have been proposed with different stages and phases defined. Differences lie on the scope, scale, and focus of each model. An agreed set of activities can be found and include analysis, design, and implementation. The classic Waterfall life cycle model, created in the 1970's by Royce and later refined by Boehm in 1976, was the first formal life cycle model where a fundamental set of development phases are defined, namely, analysis, design, programming, testing, implementation, operation, and maintenance.

Other process models include the evolutionary model, the two-leg model, and the rapid prototyping model. Evolutionary model focuses on one step at a time refinement. Two-leg model is used to develop decision-making information systems. Rapid prototyping model is similar to the Boehm's spiral model with a focus on prototyping and verification in iteration.

The Boehm's spiral model evolves from the classic Waterfall model. It is composed of a similar set of basic information systems development phases. Each phase proceeds in spiral instead of in sequence. Any number of loops can occur in a spiral. Each loop represents one activity of analysis and design. Rapid prototyping and risk management are used throughout the spiral. The order and transition among phases is based on the risk research and resolution. Boehm's spiral model is two-dimensional. The radial dimension accounts for the cumulative development costs. The angular dimension delineates the order and progress of each activity of task such as analysis and design. Recursive and sequential loops are accommodated. However, the parallel development progress needs multiple spirals to represent. Development iterations are supported but will interrupt the regular spiral sequence [Mills et al 1986].

In this paper, we adopt the Boehm's spiral model and adapt the box-structured spiral model to define requirements analysis as an incremental and evolutionary process of iterating requirements definition. We refine the model to be a process of recursive requirements recognition and refinement. The goal is to define a requirements specification of precisely stated properties and constraints that a systems must satisfy such as the scope, scale, objective, context, characteristics, constraints, assumptions, and boundaries of the proposed information systems. A recursive process model of requirements acquisition, articulation, and analysis is developed that is composed of three main activity cycles of stages. These cycles are the cycle of investigation, the cycle of specification, and the cycle of implementation that can reoccur at any point in time in the entire spiral. Each individual activity in the process model can be invoked by itself or revisited by other activities as shown in Figure 1. Nine activities are defined in the process model to describe the micro life cycle of requirements analysis. These activities consist of domain analysis, feasibility study, requirements gathering, requirements modeling, requirements specification, requirements evaluation, review and acceptance, requirements management, and requirements application.

The incremental and iterative process model of nine requirements activities is used to develop the simple set representation steps described in the paper. This representation model is based on and extended from the box structure method. It is semi-structured and hierarchical analysis and design approach. We model a piece of requirement into a piece of standard interface of stimulus-set, response-set, and state-set component. This scheme facilitates the repository and reuse capability. We extend the concept of requirements components to be organized into the requirements set model in a hierarchy. We start from the domain analysis, requirements analysis, and requirements specification. The process is feedback as well as feed forward. We feedback new details into the requirements set and evolve the requirements hierarchy from bottom up. We feed forward new discoveries into the requirements components and update the requirements set from top down. Incremental and iterative analysis activity continues and reoccurs. Our method has to follow the activities of requirements management and application. The

task of each activity in the incremental and iterative process model of requirements analysis is described as follows. These nine activities evolve from the research results presented in the general requirements determination process model of [Thayer and Dorfman 1990], the box structure methodology with object extension of [Mills, Linger, and Hevner1986] [Mills 1988] and the clean room software engineering of [Prowell, Trammell, Linger, and Poore 1999].

(1) Domain analysis activity is to identify the target systems domain and its associated environment.
(2) Feasibility study activity is to assess the economic, equipment, time, technical, personnel, organizational, legal, and development feasibilities.
(3) Requirements gathering activity is to investigate and extract the user and system's requirements, both functional and non-functional.
(4) Requirements modeling activity is to acquire, articulate, analyze, and abstract the target problem space into a model.
(5) Requirements specification activity is to record and represent requirements in a technical format.
(6) Requirements evaluation activity is to measure and evaluate the quality of requirements in terms of validity, closure, completeness, correctness, consistency, currency, compact, and clarity.
(7) Review and acceptance activity is to secure feedback from users and agreement from management.
(8) Requirements management activity is to perform the following tasks:

   a. Representation – to formalize requirements in a language.
   b. Organization – to classify, structure, and store requirements.
   c. Manipulation – to utilize and operate on requirements.
   d. Maintenance – to maintain requirements and preserve data integrity.

(9) Requirements application is to perform the following tasks:

   a. Repository – to build reusable requirements library.
   b. Reusability – to acquire requirements reusability and support the reuse of requirements.
   c. Prototyping – to examine the prototyping technique and reinforce requirements performance.

A SET MODEL OF FUNCTIONAL REQUIREMENTS REPRESENTATION

A Box-Structured Requirements Hierarchy

Following the process model, we can start each activity in sequence. Later on, depending upon the results of prior activities, we may determine if iteration is necessary to revisit a certain stage in order to refine information. A cycle of activities may be visited at the same time. If so,

the control will be transferred to that stage or that cycle and initiate the recursion. Also, we can reiterate an activity triggered by the main (macro) system development life cycle as shown in Figure 1 and Figure 2.

Figure 1. Incremental and Interative Requirements Determination Process Model

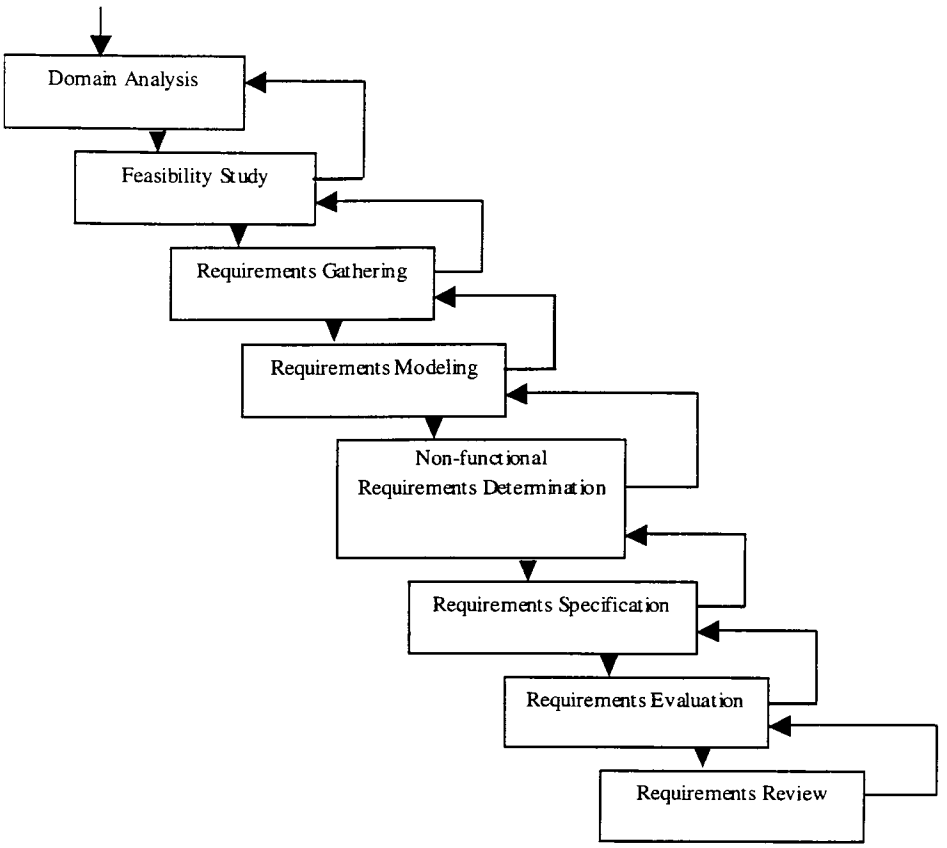
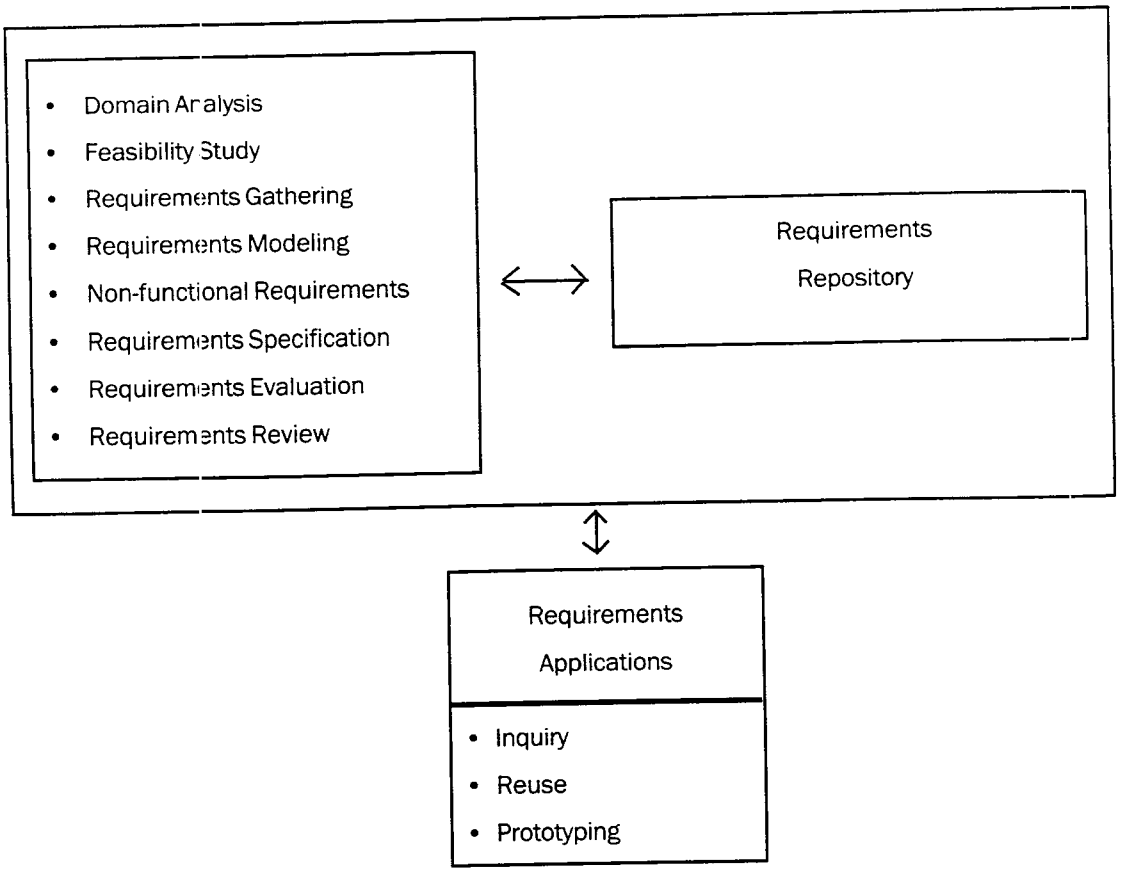
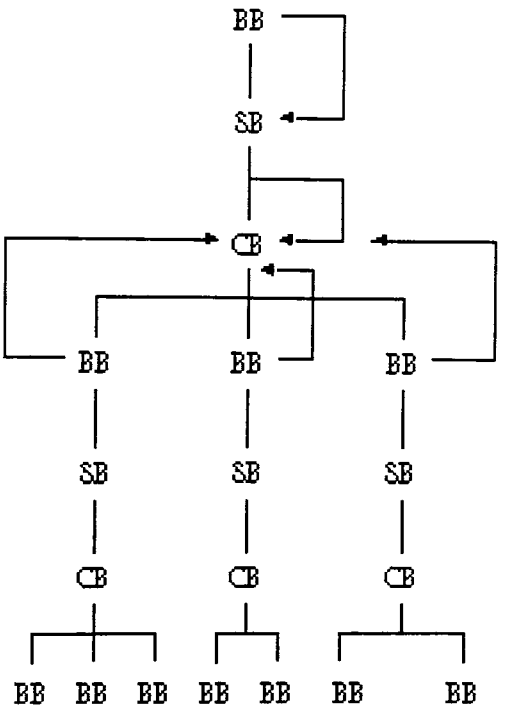


Figure 2. Block Diagram of Computer-Assisted Requirements Engineering (CARE)



In this paper, we present a hierarchical requirements analysis method that is box-structured and provides a set of techniques and procedures to assist analysts in acquiring and organizing requirements information. This set of techniques and procedures is based on the box structure hierarchy (BSH) [Mill et al 1986] [Mill 1988] and the mathematic operation of set and function that are used to define the relationships and structures among requirements information. The box structure hierarchy follows the model of stimulus, response, state, and transaction to define a piece of requirements information, i.e. an atomic requirements component with input, output, and process to be developed in the proposed systems. Three types of requirements components are defined. They are the black box (BB) requirements component, the state box (SB) requirements component, and the clear box (CB) requirements component as shown in Figure 3. Black box defines the high-level input and output specification. State box describes the data definition. Clear box details the procedurality and heuristics of functional and non-functional requirements and constraints.

Figure 3. Box Structure Hierarchy



In the requirements hierarchy, a transaction represents the basic unit of functional and non-functional requirements that accepts stimulus and produces response that in turn defines the transaction matrix. The matrix is a requirements gathering technique that tabulates transactions with stimulus, response, and state as functions and constraints. A transaction set is the universal set. Its functional set depicts the data set and process set. Non-functional set delineates the functional set, stimulus set, and response set. The core of both lies upon the data set and process set as shown in Figure 4 and Figure 5.

From the transaction matrix, we have the following sets:

Transaction Set	$T = \{T_1, T_2, ..., T_t\}$
Stimulus Set	$S = \{S_1, S_2, ..., S_s\}$
Response Set	$R = \{R_1, R_2, ..., R_r\}$
Data Set	$D = \{D_1, D_2, ..., D_d\}$
Process Set	$P = \{P_1, P_2, ..., P_p\}$
Functional Requirements Set	$F = \{F_1, F_2, ..., F_f\}$
Non-Functional Requirements Set	$N = \{N_1, N_2, ..., N_n\}$

where t, s, r, d, p, f, n are position integers

Figure 4. Data and Process Matrix

Process Matrix

Process Serial Number	Process Description and Logic
P1	<P1: Name and Logic>
P2	<P2: Name and Logic>
....	....
P_P	<P_P: Name and Logic>

Data Matrix

Process Serial Number	Process Description and Logic
D1	<D1: Name and Logic>
D2	<D2: Name and Logic>
....	....
D_d	<D_d: Name and Logic>

Figure 5. Requirement Matrix

Data Rows	Process Columns				
	P1	P2		...	P_P
Dinput(1)					
Dinput(2)					
...					
Dinput(d)					
Douput(1)					
Douput(2)					
...					
Douput(d)					

Functional Requirements Matrix

Process set based on process list is a simple listing of processes that have been identified and articulated in the domain and feasibility study. A serial number is assigned to each process, i.e. P_{number}, and a process name, i.e. <process name>, with a brief process description and logic, i.e. <process description/logic>. Data set based on data list is another listing of data item for state. A serial number, i.e. D_{number}, is given to each data item discovered, with the data name, i.e. <data name>, and with a brief data description, i.e. <data description>. These data items are further divided into inflows and outflows to distinguish data of incoming and outgoing information. Di and Do notations are used. Data inflow corresponds to stimulus set and data outflow corresponds to response set.

Applying the process list and data list, analysts further build the requirements matrix that is a focused two-dimensional table to tabulate data set and process set. Each column is a process and each row is a data item of inflow and outflow. The matrix is prepared to identify and mark the data elements involved in each process column then cross-mark process items for each data. The complete requirements matrix is shown in Figure 6.

In the requirements matrix, we have the following sets.
Process Set $P=\{P_1, P_2, \ldots, P_p\}$
Data Inflow$=\{Di_1, Di_2, \ldots., Di_{di}\}$
Data Outflow$=\{Do_1, Do_2, \ldots, Do_{do}\}$
Where p, di, do are position integers, and P_{numbe} is a simple and mathematical function describing logic as relationships between domain and co-domain to be one-to-one, one-to-many, many-to-one, and many-to-many mapping.

A Formulation of Requirements Hierarchy

Each process column with data elements represents a requirements component that contains information on the process logic and data definition. Using sets and functions, their relationships are defined and structured into a hierarchy. We develop a set of formulation rules to guide the development of the hierarchy of requirements components. Rules are classified into rules of level identification and rules of addition and deletion element.

(1) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$

Then Px is the parent component of Py and Pz

(2) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$ and

Figure 6. Transaction Matrix

Requirements Component	Transactions							
	T_1	T_2	...	T_t	C_1	C_2	...	C_c
Stimulus S_1 S_2 ... S_S								
Response R_1 R_2 ... R_r								
Function F_1 F_2 ... F_f								
Data D_1 D_2 ... D_d								
Process P_1 P_2 ... P_p								
Non-Functional Requirements N_1 N_2 ... N_n								

Notations:
- T_{number}: Transaction Serial Number, where number 0 {1, ..., t}
- S_{number}: Stimulus Serial Number, where number 0 {1, ..., s}
- R_{number}: Transaction Serial Number, where number 0 {1, ..., r}
- F_{number}: Function Serial Number, where number 0 {1, ..., f}
- D_{number}: Data Serial Number, where number 0 {1, ..., d}
- P_{number}: Process Serial Number, where number 0 {1, ..., p}
- N_{number}: Non-Functional Requirements Serial Number, where number 0 {1, ..., n}

t, s, r, f, d, p, n are positive integers

$P_y \cap P_z =$ and

$DI_{pt} \not\subset Do_{pz}$

Then P_y and P_z are at the same level.

(3) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$ and

$P_y \cap P_z = \phi$ and

$Di_{pz} \not\subset Do_{py}$

Then P_y and P_z are at the same level.

(4) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$ and

$P_y \cap P_z = \phi$ and

Then P_y and P_z can be at the same or at different levels.

(5) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$ and

$P_y \cap P_z = \phi$ and

$P_y \not\subset P_z$ and

$P_z \not\subset P_y$

Then P_y and P_z can be at the same or different levels.

(6) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$ and

$P_y \cap P_z = \phi$ and

$P_y \subset P_z$

Then P_z is the parent component of P_y.

(7) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$ and

$P_z \cap P_z = \emptyset$ and

$P_z \subset P_y$

Then P_y is the parent component of P_z.

(8) If $P_y \subset P_x$ and

$P_z \subset P_x$ and

$P_y \neq P_z$ and

$P_y \cap P_z =$ and

$P_y \subset P_z$ and

$P_z \subset P_y$

Then P_y and P_z are identical.

(9) Adding New Requirements components, Rx,

If new data inflow $DI_x \in$ Di, new data outflow $Do_x \in$ Do,

Then add a new process column Px into the requirements matrix,

i.e. $P_{p+}1 <- P_x, p <- p+1$, and follow the above rules to position R_x

Else add a new process column P_x into the process matrix, i.e.

$P_{p+}1 <- P_x, p <- p+1$, and add $Di_{Di+1} <- Di_x$ or add $Do_{D0+1} <- Do_x$

and follow the above rules to position Rx.

(10) Deleting existing requirements components, Rx,

If $\forall x \in Di_{pi}, \exists x \in \sim Di_{pi}, \forall y \in Do_{pi}, \exists y \in \sim Do_{pi}$,

Then remove the process column P_i from the requirements matrix,

i.e. $P_{p-1} <- P_p$, $p < p-1$, and remove Di_{pi}, $Di_{Di-1} <- Di_{Di}$, $D_i <- D_{i-1}$,

and remove Do_{pi}, $Do_{DO-1} <- Do_{Do}$, $Do <- Do-1$, and remove R_x

Else remove the process column P_i from the requirements matrix, i.e. $P_{p-1} <- P_p$, $p <- p-1$

An Evolution of Requirements Hierarchy

A requirements hierarchy evolution is developed and expanded from process set and data set. The evolution is managed and represented in set and logic operation. Since the hierarchy is a usage hierarchy of transactions to display the relationship and interaction between requirements components. Based on the commonality of process set and data set to offer the functional

cohesiveness and non-functional relatedness, requirements components evolve top-down and bottom-up. Top-down decomposition procedure allows the requirements components hierarchy to be expanded into smaller and simpler requirements sub-components. Bottom-up composition procedure starts from the sub-components even the sub-hierarchies then connects them into the main hierarchy that in turn defines their upper-level components. The hierarchy evolution rules are classified into rules of level changes and rules of transaction composition/decomposition to be described as follows.

(1) The definition of a requirement's component is expanded as follows:

requirements component = $T_x = S_{Tx} \cup R_{Tx} \cup F_{Tx}, S_{Ti} \subseteq S, R_{Ti} \subseteq R, F_{Ti} \subseteq F$

(2) If $T_y \subseteq T_x, T_z \subseteq T_x$, and T_y and $T_y \neq T_z$, where x, y, z $\in \{1,2,...,t\}$, then T_x is a requirements component at the higher level and T_y and T are at the lower levels under T_x, and T_y and T_z can be at the same or different levels. This is determined in (3) and (4). And all the T_i's where i $\in \{1,2,...,t\}$ under consideration are subsets of the largest set, called the universe set which is the requirements component at the root of the requirements hierarchy. No complete set relations can be found between T_4 and T_3 so they can be at the same or different lower levels under T_1, since $S_{T5}, R_{T5} \not\subset S_{T1}, R_{T1}$, and no parent-child relation exists between them.

(3) If $T_y \neq T_z$, $T_y \cap T_z \neq \phi$, T_y and $T_z \subset T_x$, where x, y, z $\in \{1,2,...,t\}$, then T_y and T_z are two requirements components under the larger black box of T_x and at different levels.

(4) If $T_y \cap T_z = \phi$, T_y and $T_z \subset T_x$, where x, y, z $\in \{1,2,...t\}$, then T_y and T_z are two requirements components under the larger requirements component of T_x and at different levels.

(5) In general, if $T_x : S_{Tx} \rightarrow R_{Tx}, S_{Tx} \in U$, where $U = \{U_1, U_2, ..., U_u\}$, $R_{Tx} \in V$, where $V = \{V_1, V_2, ..., V_v\}$ and there are $T_{y1}, T_{y2}, ..., T_{yn}$, where u, v, n are positive intergers, and

If $S_{y1}, S_{y2}, ..., S_{yn} \subset U$ or

$S_{y1}, S_{y2}, ..., S_{yn} = U$ or

$R_{y1}, R_{y2}, ..., R_{yn}, = V$ or

Then $T_{y1}, T_{y2}, ..., T_{yn}$ are at the same level and under the requirements component T_x.

A SET MODEL OF NON-FUNCTIONAL REQUIREMENTS REPRESENTATION

Non-functional requirements are a set of expectations and conditions imposed on the target systems by the users, management, developers, and environment. If the functional requirements set determines what the systems should be, then the non-functional requirements is a set of descriptions to define how the systems should behave. Non-functional requirements can assure that the systems to be built will be achievable and acceptable. From the literature review, we compile a general list of five high-level categories of non-functional requirements and con-

straints. These categories consist of the design constraints, the performance requirements, the systems requirements, the implementation constraints, and the managerial consideration [Nixon 1993] [Chung, Nixon, and Yu 1995] [Robertson and Robertson 1999] and [Nuseibeh, Easterbrook, and Russo 2001]. [Nixon 1993] [Chung, Nixon, and Yu 1995] describe the performance requirements analysis needed in ISD and change management. [Robertson and Robertson 1999] details the non-functional requirements contents from the business process and practice perspective. [Nuseibeh, Easterbrook, and Russo 2001] discusses the consistency and completeness requirements analysis that needs rule base and frame base to represent these non-functional requirements.

Non-functional requirements can be quantified such as the common performance requirements and implementation constraints. Some important non-functional requirements cannot be quantified such as design constraints and managerial considerations. It is our intent to provide a set logic and mathematic expression to represent the quantifiable and non-functional requirements. We intend to treat a non-functional requirement information as a component that can be represented as stimulus-set, response-set, and state-set. And, in terms of the non-quantifiable and non-functional requirements, we address them with syntax and grammar to represent their rules and examples. This simple set representation and syntactic representation of non-functional requirements can achieve the one unifying model scheme. Consistency and cohesiveness are attained through the set logic and operation. We describe each category of non-functional requirements as follows. Details are listed in Appendix A.

(1) Design constraints mean the data constraints that are imposed on the use of data, the flow of data, and the distribution of data; and the process constraints that are imposed on the use of process, the sequence of process, and the limitation of process.
(2) Performance requirements mean the subjective and objective anticipations placed on the proposed systems, such as the response time, throughput, reliability, memory allocation, and usage. Sub-categories are further defined to include time, cost, space, utilization, availability, reliability, survivability, and security.
(3) Systems requirements mean the effectiveness and efficiency of strategies and methodologies to be used in the systems development, such as the flexibility, transferability, maintainability, and cost benefit analysis. Sub-categories are further defined to contain the system scope and scale, maintainability, change control and management, and quality control and management.
(4) Implementation requirements mean the rules and procedures that are used to conduct the activities of programming, testing, conversion, training, and documentation.
(5) Managerial requirements mean the organizational, legal, economic, and behavioral considerations and constraints imposed on the systems.

With the above classification, we next present a formalization of non-functional requirements into box-structured components with stimulus set and response set. A simple set model is created to organize and structure the hierarchy of non-functional requirements. This model

originates from the idea of the set logic and operation in mathematics. We acquire non-functional requirements and articulate property in a representation set logics and language.

First, we define a non-functional requirements set to be comprised with a pair of sets of input (stimulus) and output (response). The basic element of a set is a unit of input, output, data, process, or constraint. Set and logic operation, algorithm, and grammar, are adopted to represent the quantitative and qualitative non-functional requirements. Usually, design constraints, performance requirements, and systems consideration are more quantifiable. Set and logic operation and algorithm are applied. Implementation constraints and managerial consideration are less quantifiable. Grammar and syntax are applied. We describe the use of application as follows.

Symbolic Logic

Set and logic symbols along with their operations are used to represent predicates, relationships, and Boolean expressions.

(1) Set and Logic Symbols

Function set, F_x, who's input set is I_{Fx}, output set is O_{Fx}, is used to define the unit of non-functional requirements and constraints, we use D_{Fx}, and P_{Fx} to represent data and procedural constraints.

For instance:

$D_{Fx} = \{\forall a \in I_{Fx}, a>0\}$

$P_{Fx} = \{\exists b \in F_{Fx}, b \text{ is recursive}\}$

(2) Set and Logic Operations

In some cases, the cross product of sets occurs and a sequence of elements results due to set and logic operation. We use the symbolic operations to represent them.

For instance:

For $A \subset I_{Fx}$, $B \subset I_{Fx}$, and $A \cap B = \phi$, Then $D_{Fx} = A \times B$

Algorithms

Algorithms in the form of control constructs are used to define the flow, conditions, and control of elements of non-functional constraints.

For instance:
Sequence: $D_{Fx(F1;F2;F3)}$
Alteration: $D_{Fx(F1|F2|F3)}$
Iteration: $D_{FX(F1;F2)^*}$

Formal Grammars

Language syntax provides the means to integrate and represent of non-functional requirements in specifics and statements based on requirements set.

For instance:
<non_functional_requirements>
::= begin non-functional-requirements;
<data_constraint>
|<procedural_constraint>
|<performance_constraint>
|<economic_constraint>
|<design_constraint>
|<managerial_constraint>
<data_constraint>
::= begin data constraint;
<data_constraint>
$D_{Fx}=\{I_{Fx} | O_{Fx}\}\{;|?|\&\}^*\{I_{Fx} | O_{Fx}\}$;
$| D_{Fx}$ =<set_expr>;
end data constraint;
<procedural_constraint >
::= begin procedural_constraint;
$P_{Fx}=\{Fi\,(I_{Fi} | O_{Fi})|^*Fj(I_{Fj} | O_{Fj})\}\ \{;|?|\&\}^*\{Fi(I_{Fi} | O_{Fi})|^*Fj(I_{Fj} | O_{Fj})\}$;
$|P_{Fx}$ =<set_expr>;
end procedural_constraint;

Quality Requirements Representation

Quality evaluation is a process of analyzing and assessing the degree and extent of the desired quality possessed by the software design. The primary set of quality attributes includes completeness, correctness, consistency, comprehensiveness, and clarity [Bastani, DeMarco, Pasquini 1993] [Briand, Thomas, Hetmanski 1993] [Russo, Nuseibeh, and Kramer 1998]. A brief definition of each criterion is given as follows.

(1) Completeness means data and procedure closure within the requirements specification.
(2) Correctness means accuracy and integrity of data and functions against the requirements specification.
(3) Consistency means reference and regularity of requirements cross-checked through the requirements specification.
(4) Comprehensiveness means understandability and informativeness of requirements without sacrificing conciseness.
(5) Clarity means there is no ambiguity. Language and graphic representation are in syntax and terms.

For instance, in completeness, every input needed for a function should have been defined in the input set. Every output resulted from a function should have been defined in the output set.

For instance:

A function set, Fi, with its input subset, I_{Fi}, and output subset, O_{Fi}, that $O_{Fi=F(I}$ i Fi), and right hand side (RHS) of F_i is I_{Fi} and left hand side (LHS) of Fi is O_{FI}; $I_{Fi} \subseteq I\, I_{Fi} = \{I_{Fi}1, I_{Fi}2, ..., I_{Fi}n\}$, where n is a positive integer

Then

(1) $\forall I_i \in \{\text{RHS of Fi}\} = > I_i \in I_{Fi}$

(2) $(I_i \in \{\text{RHS of Fi}\} \wedge I_i \notin I_{Fi}) \vee (I_i \notin \{\text{RHS of Fi}\} \wedge I_i \in I_{FI} = \text{FALSE}$ and $O_{FI} \subseteq O\, O_{FI} = \{O_{Fi}1, O_{Fi}2, ..., O_{Fi}m\}$, where n is a positive integer

Then

(i) $\forall O_i \in \{\text{LHS of Fi}\} = > O_i\, O_{Fi}$

(ii) $(O_i \in \{\text{LHS of Fi}\} \wedge O_i \notin O_{Fi}) \vee (Oi\ \{\text{LHS of Fi}\} \wedge O \in O_{FI}) = \text{FALSE}$

Every function that is needed to accept an input and to produce output should have been defined in the function set, i.e. for this case, $F_x = I_x \cup O_x \cup D_x \cup P_x$, where x is a positive integer, and $F_x \subset F, D_x \subset D, P_x \subset P$, without considering $I_x, I_x \subset I$ and $O_x, O_x \subset O$, because we are testing if sufficient functions are defined for every input and output.

For instance:

For a is an input then

(1) $\forall a \in I_x = > a \in (F_x \cup D_x \cup P_x)$

(2) $(a \in I_x \wedge a \in (F_x \cup D_x \cup P_x) \vee (a \in I_x\ A\ (F_x \cup D_x \cup P_x) = \text{FALSE}$

For b is an output then

(i) $\forall b\ O_x = > b \in (F_x \cup D_x \cup P_x)$

(ii) $(b \in O_x \wedge a \notin (F_x \cup D_x \cup P_x) \vee (b \notin I_x \wedge b \in (F_x \cup D_x \cup P_x) = \text{FALSE}$

For every function Fx, the input, I_x, should appear on the left hand side (LHS) of the formula, and the ouptut, O_{FX}, should appear on the right hand side (RHS) of the formula.

For instance, in correctness, use the "Isolation Programming Technique" to check every function in terms of the correctness of input and output.

For instance:
F_X: I_X-> O_X, where $I_X = \{I_X1, I_X2, \ldots, I_Xn\}$
For I=1....n
Loop
I_i=(value);
I_k=0;(*k≠I, k∈ {1...n}*) evaluate O_X
End Loop

A SIMPLE EXAMPLE

A simple and real world case study is developed to illustrate the basic usage of this method. We follow the requirements determination process model and demonstrate the use of the requirements matrix, transaction matrix, box structure hierarchy, and box structure analysis. Functional and non-functional information are collected and captured. Due to the space limit, we only describe the first level of requirements representation.

The case study is a requirements acquisition and analysis task of a Job Placement Center at a Midwest State University. In the simple pilot experiment, we use this case as the experimental task and we prepare the requirements representation in detail at the first level for illustration.

The job placement center at the Midwest State University offers placement and counseling services to students and alumni. The center supports approximately 9500 registrants annually. Its primary purpose is to help individuals develop career planning. To this end, it offers five services: (1) placement counseling and referral, (2) career counseling, (3) a weekly publication of employment vacancies, (4) an on-campus interview program, and (5) campus and community relations activities. The system to be analyzed and designed will mainly support the placement registration, counseling, referral, job vacancy publication, and on-campus interview program.

First Step:

We use the transaction matrix to collect the first level of data and transaction requirements. Transaction list tabulates the main functional and non-functional requirements as shown below. Data list tabulates the input and output data requirements based on our interviews, surveys, and documents.

Job Placement Center Transaction List

Transaction No.	Transaction Description
T_1	Job Placement Services
T_2	Placement Registration Process
T_3	Placement Counseling and Review
T_4	Placement Filing Process
T_5	Receive Registrants' Data
T_6	Setup Placement Folders
T_7	Setup Placement Binders
T_8	Setup Card Reference
T_9	Registration Confirmation
T_{10}	Placement Maintenance Process
T_{11}	Receive Placement Labels
T_{12}	Sort Placement Labels
T_{13}	Update Placement Folders
$T_{14...T40}$	

Job Placement Center Data List

Data No	Data Name	Data Description
A_1	Placement Packets	Instructions
		Consent Form
		Data Sheet
		Data Sheet Worksheet
		Fee Statement
		Mailing Instructions
		Placement Center Fact Sheet
A_2	Placement Folders	Consent Form
		Data Sheet
		Fee Receipt
A_3	Updated Placement	Recommendation Letters
		Instructor Evaluation
		Student Teaching Evaluation

		Transcripts
		Resume
		Consent Form
		Data Sheet
		FeeReceipt
A_4	Placement Binders	Data Sheet
A_5	Placement Card Reference	
A_6	Confirmation	Welcome Letter
		Request of Position
		Secured Form
		Six Placement Labels
A_7	Registrants' List	
A_8	Inactive Placement	Position Secured Form
		Recommendation Letters
		Instructor Evaluation
		Student Teaching Evaluation
		Transcripts
		Resume
		Consent Form
		Data Sheet
		Fee Receipt
A_9	Employer Folders	
A_{10}	Updated Employer Position Secured Form	
A_{11}	Inquiry Letter	
A_{12}	Employer Mailing List	
A_{13}	Inactive Employer Folders	
A_{14} ... A_{23}		

Second Step:

Each transaction become an element in the transaction set. The set will be analyzed with stimulus, response, and state-machine scheme to formulate a set of hierarchical transaction sets.

Stimulus Set

S_1	Placement Service Requests
S_2	Placement Packets
S_3	Reviewed Data Sheet and Consent Form
S_4	Registration Fee Receipts
S_5	Placement Folders
S_6	Recommendation Letters
S_7	Instructor Evaluation
S_8	Student Teaching Evaluation
S_9	Transcripts
S_{10}	Resume
S_{11}	Services Fee Receipts
S_{12}	Placement Binders
S_{13}	Placement Card References
$S_{14 \ldots S32}$	

Response Set

R_1	Placement Service Requests
R_2	Placement Packets
R_3	Reviewed Data Sheet and Consent Form
R_4	Registration Fee Receipts
R_5	Placement Folders
R_6	Recommendation Letters
R_7	Instructor Evaluation
R_8	Student Teaching Evaluation
R_9	Transcripts
R_{10}	Resume
R_{11}	Services Fee Receipts
R_{12}	Updated Placement Binders
R_{13}	Updated Placement Card Reference
$R_{14\text{-}R32}$	

Third Step:

We form the transaction set, the stimulus set, and the response set to give the set-centric box structure hierarchy as shown below. Only a sample of functional and non-functional requirements is described here.

$T = \{T_x | x?(1,2, ..., 40)\}$

$T =$

$\{T_1,T_2,T_3,T_4,T_5,T_6,T_7,T_8,T_9,T_{10},T_{11},T_{12},T_{13},T_{14},T_{15},T_{16},T_{17},T_{18},T_{19},T_{20},T_{21},T_{22},T_{23},T_{24},T_{25},T_{26},T_{27},T_{28},T_{29},T_{30},T_{31},T_{32},T_{33},T_{34},T_{35},T_{36},T_{37},T_{38},T_{39},T_{40}\}$

T1

$S_{T1} =$

$\{S_1,S_2,S_3,S_4,S_5,S_6,S_7,S_8,S_9,S_{10},S_{11},S_{12},S_{13},S_{14},S_{15},S_{16},S_{17},S_{18},S_{19},S_{20},S_{21},S_{22},S_{23},S_{24},S_{25},S_{26},S_{27},S_{28},S_{29},S_{30},S_{31},S_{32}\}$

$R_{T1} =$

$\{R_1,R_2,R_3,R_4,R_5,R_6,R_7,R_8,R_9,R_{10},R_{11},R_{12},R_{13},R_{14},R_{15},R_{16},R_{17},R_{18},R_{19},R_{20},R_{21},R_{22},R_{23},R_{24},R_{25},R_{26},R_{27},R_{28},R_{29},R_{30},R_{31},R_{32}\}$

T_2

$S_{T2} = \{S_1\}$

$R_{T2} = \{R_2,R_3,R_4\}$

P_{T2} = Placement Packet Information Must Be Completed and Reviewed
Counseling Session Must Be Attended
Registration Fee Must Be Paid
To Be Considered Registered Registrants

N_{T2} = Students and Alumni Only

T_3

$S_{T3} = \{S_1\}$

$R_{T3} = \{R_2,R_3,R_4\}$

F_{T3} = Receive Placement Services Request
Provide Placement Packet
Provide Counseling Session
Review Placement Packet Information
Receive Registration Fee

T_4

$S_{T4} = \{S_3,S_4\}$

$R_{T4} = \{R_3,R_4,R_5,R_{12},R_{13},R_{14},R_{15}\}$

T_5

$S_{T5} = \{S_3,S_4\}$

$R_{T5} = \{R_3,R_4\}$

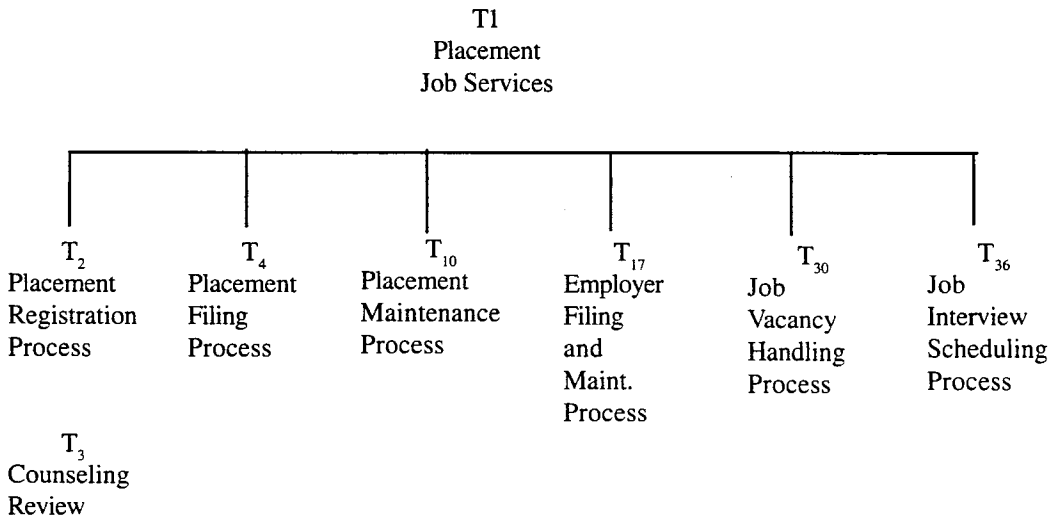
F_{T5} = Registrants' Information Forwarded from the Counseling
Registrants' Information Received by Mail

...

T_{11}
$S_{T11} = \{S_{16}\}$
$R_{T11} = \{R_{16}\}$
F_{T11} = Receive Returned Placement Label to Maintain the Placement Folder
P_{T11} = Registrants Must Return the Monthly Placement Label Due by 10th of Each Month at the Center

T_{12}
$S_{T12} = \{S_{16}\}$
$R_{T12} = \{R_{16}\}$
F_{T12} = Sort Returned Placement Label to Maintain the Placement Folder
P_{T12} = Sorting Is Performed on the 20th of Each Month

T_{13}
$S_{T13} = \{S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{16}\}$
$R_{T13} = \{R_{18}\}$
F_{T13} = Update the Placement Folders with the Sorted and Returned Placement Labels
Insert Additional Registrants' Information into the Placement Folders
P_{T13} = If Services Are Requested, Service Fees Must Be Paid



DISCUSSION AND LIMITATION

This research develops an incremental and evolutionary requirements determination model to support an iterative and recursive process of requirements acquisition, analysis, and annotation. A simple set model is created based on the set logic and the box structure hierarchy to support a complete set of requirements activities and relationships defined in the spiral process model. The set model creates a requirements hierarchy to be described in arithmetic representation and stimulus-state-response structure. An integrated and incorporated requirements matrix tackles functional and non-functional requirements and constraints.

Problem space and solution space are separated. Requirements analysis specification and design alternative specification are divided. The focus is on the development of a mathematical and methodological approach that can determine and denote requirements and constraints in the problem domain without specifying the logical or physical design resolutions. In the method, theory of set and function are used. Commonality is applied to support reuse. Validation and verification is facilitated top-down and bottom-up via the logic operation.

In the model, abstraction of a requirements is in a box representation of stimulus-state-function-response set. It starts from data matrix and process matrix. Relationship of requirements box is in a hierarchy representation of decomposition and composition logic. It begins from requirement matrix and transaction matrix. It is the aim of this study to present a breakthrough in tackling the main technical and behavioral obstacle, i.e. to produce a tractable and transitional set of requirements. However, our method has its limitations. The stimulus-response-state-function model aims to capture and compile requirements that have input-output-data-process property. Requirements without the property cannot be modeled in this method. Further, the set representation and revelation aims to express and extract logic and heuristic in the property. Requirements without the property cannot be modeled in this method.

As discussed in the literature review, our set model is not knowledge-based. Hence it differs from [Hudlicka 1996]. This method focuses on presenting one integral set approach to model and manipulate requirements and constraints. Therefore, it differs from a practical prototyping approach [Robertson and Robertson 1999]. Further, we perceive requirements determination to be a process that is incremental and iterative so as to develop a mathematic requirements representation into a structure It differs from [Boehm, Bose, Horowitz, and Lee 1994] [Leveson 1998] that adopt the concept of executable specification from the requirements. More, we did not apply scenario and use case from the business process but leave domain analysis as given. Hence, our method differs from [Gough, Fodemski, Higgins, and Ray 1995] [Rawsthorne 1996] and [Nissen, Jeusfeld, Marke, Zemanek, and Huber 1996]. However, we treat perspectives as one requirements box with views defined in terms of stimulus, state, response, and function that is similar to the multiple viewpoint method described in [Finkelstein, Gabbay, Hunter, Kramer, and Nuseibeh 1994] [Nuseibeh, Kramer, and Finkelstein 1994]. The rationale is to model functional cohesiveness and non-functional relatedness in set at the beginning.

CONCLUDING REMARKS AND FUTURE RESEARCH DIRECTIONS

Requirements analysis is the most difficult area to automate under the computer-aided software engineering (CASE) environment. Methodological and mathematical approaches are needed to tackle the fundamental and structural challenges. We propose a incremental and iterative process model as the first step to develop a more formal and systematic approach. As a second step, we present a simple and hierarchical set logic and operation to represent a number of types of requirements components. Following that, we propose an alternative to examine the quality of the represented requirements components in terms of completeness, correctness, and consistency. We present an alternative to recruit and represent a tractable and transitional set of functional and non-functional requirements.

In the future research directions, we will be looking at an implementation of a requirements component editor, a requirements representation generator, and a reuse repository to develop a computer-aided requirements engineering environment (CARE). We hope the base of the structure is built on the simple set model of logic and operation for functional and non-functional requirements. However, more work needs to be done on the non-functional requirements refinement and revision. We intend to develop an integral crosschecking scheme between functional and non-functional requirements sets to reinforce the validation and verification of quantitative and qualitative requirements and constraints.

REFERENCES

[Acosta, Burns, Rzepka, and Sidoran 1994] Acosta, R. D., Burns, C. L., Rzepka, W. E., and Sidoran, J. L., "A Case Study of Applying Rapid Prototyping Techniques in the Requirements Engineering Environment," The First International Conference on Requirements Engineering, April 1994, pp. 66-73.

[Bastani, DeMarco, and Pasquini 1993] Bastani, DeMarco, and Pasquini, "Experimental Evaluation of a Fuzzy Set Assessment of Software Correctness Using Programming Mutation," Proceedings of International Conference on Software Engineering, 1993.

[Boehm 1988] Boehm, R. W., "A Spiral Model of Software Development and Enhancement," IEEE Computer, May 1988, pp.61-72.

[Boehm, Bose, Horowitz, and Lee 1994] Boehm, B., Bose, P., Horowitz, E., and Lee, M. J., "Software Requirements as Negotiated Win Conditions," The First International Conference on Requirements Engineering, April 1994, pp. 74-83.

[Bolaka 1997] Bolaka, Requirements Engineering, John Wiley and Sons Co., 1997.

[Borgida, Greenspan, and Mylopoulos 1985] A. Borgida, Greenspan, S., and Mylopoulos, J. "Knowledge Representation as the Basis for Requirements Specification," IEEE Computer, pp.82-91, April 1985.

[Brian, Thomas, and Hetmanski 1993] Brian, Thomas, and Hetmanski, "Modeling and Managing Risk Early Software Development" Proceedings of International Conference on Software Engineering, 1993.

[Cardenas and Zelkowitz 1990] Cardenas, S. and Zelkowitz, M., "Evaluation Criteria for Functional Specifications," a paper appears in Systems and Software Requirements Engineering, IEEE Computer Society, 1990.

[Chung, Nixon, and Yu 1995] Chung, L., Nixon, B.A., and Yu, E., "Using Non-Functional Requirements to Systematically Support Chang," IEEE International Symposium on Requirements Engineering, March 1995, pp.132-139.

[Davis 1982] G. Davis, "Strategies of Information Requirements Determination," IBM Systems Journal, Vol. 21, No.1, pp.4-32, January 1982.

[Davis 1988] A. Davis, "A Comparison of Techniques for the Specification of External Systems Behavior," Communications of the ACM, Vol. 31, No. 9, September 1988.

[Davis 1994] A. Davis, Software Requirements Analysis and Specification, Prentice Hall, 1994.

[Emam and Madhavji 1994] Hughes, K.J., Rankin, R.M. and Sennett, C.T., "Taxonomy for Requirements Analysis," The First International Conference On Requirements Engineering, April 1994, pp. 176-179.

[Farbey and Finkelstein 2001] Farbey, B, & Finkelstein, A. "Software Acquisition: a Business Strategy Analysis", Requirements Engineering (RE 2001).

[Finkelstein 2000] Finkelstein, A., The Future of Software Engineering, Ed., ACM Press, May 2000.

[Finkelstein, Gabbay, Hunter, Kramer, and Nuseibeh 1994] Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., and Nuseibeh, B., "Inconsistency Handling In Multi-Perspective Specifications," IEEE Transactions on Software Engineering, 20, 8, (1994), 569-578.

[Goldin and Berry 1994] Goldin, L. and Berry, D. M., "AbstFinder, A Prototype Abstraction Finder for Natural Language Text for Use in Requirements Eliciation: Design, Methodology, and Evaluation," The First International Conference on Requirements Engineering, April 1994, pp. 84-93.

[Gotel and Finkelstein 1994] Gotel, O. C. Z. and Finkelstein, A. C. W., "An Analysis of the Requirements Traceability Problem," The First International Conference on Requirements Engineering, April 1994, pp.94-101.

[Gotel and Finkelstein 1995] Gotel, O. and Findelstein, A., "Contribution Structures," The Second International Symposium on Requirements Engineering, March 1995, pp. 100-107.

[Gough, Fodemski, Higgins, and Ray 1995] Gough, P. A., Fodemski, F. T., Higgins, S. A., and Ray, S. J., "Scenarios – An Industrial Case Study and Hypermedia Enhancements," The Second International Symposium on Requirements Engineering, March 1995, pp. 10-17.

[Hall, Jackson, Lane, Nuseibeh, and Rapanotti 2002] Hall, J., Jackson, M., Laney, R., Nuseibeh, B. and Rapanotti, R. "Relating Software Requirements and Architectures using Problem Frames," Proceedings of IEEE International Requirements Engineering Conference (RE'02), Essen, Germany, 9-13 September 2002.

[Hudlicka 1996] Hudlicka, E., "Requirements Elicitation with Indirect Knowledge Elicitation Techniques: Comparison of Three Methods," The Second International Conference on Requirements Engineering, April 1996, pp. 4-11.

[Hughes, O'Brien, Rodden, Rouncefield, and Sommerville 1995] Hughes, J., O'Brien, J., Rodden, T., Rouncefield, M., and Sommervile, I., "Presenting Ethnography in the Requirements Process," The First International Symposium on Requirements Engineering, January 1995, pp. 27-35.

[Jackson 1996] Jackson, M., Software Requirements Specification, Addison Wesley, 1996.

[Keller, Kahn, and Panara 1990] Keller, S., Kahn, L., and Panara, R., "Specifying Software Quality Requirements with Metrics," a paper appears in Systems and Software Requirements Engineering, IEEE Computer Society, 1990.

[Leveson 1998] Leveson, N. G., "Intent Specifications: An Approach to Building Human-Centered Specifications," The Third International Conference on Requirements Engineering, April 1998, pp. 204-213.

[Maiden, Mistry, and Sutcliffe 1995] Maiden, N. A. M., Mistry, P., and Sutcliffe, A. G., "How People Categorize Requirements for Reuse: A Natural Approach," The Second International Symposium on Requirements Engineering, March 1995, pp.148-157.

[Menzies, Easterbrook, Nuseibeh, and Waugh 1999] Menzies, T., Easterbrook, S., Nuseibeh, A. and Waugh, S. "An Empirical Investigation of Multiple Viewpoint Reasoning in Requirements Engineering", Proceedings, Fourth IEEE International Symposium on Requirements Engineering (RE'99), Limerick, Ireland, June 7-11, 1999.

[Mills. Linger, and Hevner1986] Mills, H., Linger, R., and Hevner, A., Principles of Information Systems Analysis and Design , Academic Press, Inc., 1986.

[Mills 1988] H. Mills, "Stepwise Refinement and Verification in Box-Structured Systems," IEEE Computer, Vol. 21, No. 6, June 1988.

[Nentwich, Capra, Emmerich, and Finkelstein 2002] Nentwich, C., Capra, L., Emmerich, W. and Finkelstein, A. "xlinkit: A Consistency Checking and Smart Link Generation Service" To appear: ACM Transactions on Internet Technology, 2002.

[Nissen, Jeusfeld, Marke, Zemanek, and Huber 1996] Nissen, H. W., Jeusfeld, M. A., Jarke, M. Zemanek, G. V., and Huber, H., "Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology," The Second International Conference on Requirements Engineering, April 1996, pp. 217-221.

[Nixon 1993] Nixon, B. A., "Dealing with Performance Requirements During the Development of Information Systems," The First International Symposium on Requirements Engineering, January 1993, pp. 42-49.

[Nuseibeh and Easterbrook 2000] Nuseibeh, A. and Easterbrook, S. "Requirements Engineering: A Roadmap", In A. C. W. Finkelstein (ed) "The Future of Software Engineering ", companion volume to the proceedings of the 22nd International Conference on Software Engineering, ICSE 2000.

[Nuseibeh, Easterbrook, and Russo 2000] Nuseibeh, A., Easterbrook, S. and Russo, A. "Leveraging Inconsistency in Software Development", IEEE Computer, Volume 33, No. 4, Pages 24-29, April 2000.

[Nuseibeh, Kramer, and Finkelstein 1994] Nuseibeh, B., Kramer, J. and Finkelstein, A. "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification," IEEE Transactions on Software Engineering, 20, 10 (1994), 760-773.

[Nuseibeh, Easterbrook, and Russo 2001] Nuseibeh, B., Easterbrook, S. and Russo, A. "Making Inconsistency Respectable in Software Development ", Journal of Systems and Software, 58 (2) (2001) pp. 171-180.

[Prowell, Trammell, Linger, and Poore 1999] Prowell, Trammell, Linger, and Poore, Cleanroom Software Engineering, Addison Wesley Co., 1999.

[Ramesh and Luqi 1993] Ramesh, B. and Luqi, "Process Knowledge Based Rapid Prototyping for Requirements Engineering," The First International Symposium on Requirements Engineering, January 1993, pp. 248-255.

[Rawsthorne 1996] Rawsthorne, D. A., "Capturing Functional Requirements Through Object Interactions," The Second International Conference on Requirements Engineering, April 1998, pp. 60-68.

[Robertson and Robertson 1999] Robertson, J. and Robertson, S., Complete Systems Analysis : The Workbook, the Textbook, the Answers, Addison Wesley Co., 1999.

[Robertson and Robertson 2000] Robertson, J. and Robertson, S., Mastering the Requirements Process, Addison Wesley Co., 2000.

[Russo, Nuseibeh, and Kramer 1998] Russo, A., Nuseibeh, B., and Kramer, J., "Restructuring Requirements Specifications for Managing Inconsistency and Change: A Case Study," The Third International Conference on Requirements Engineering, April 1998, pp. 51-61.

[Russo, Miller, Nuseibeh, and Kramer 2002] Russo, A., Miller, R., Nuseibeh, B. and Kramer, J. "An Abductive Approach for Analysing Event-Based Requirements Specifications," Proceedings of 18th International Conference onLogic Programming, Copenhagen, Denmark, 29 July-1 August 2002.

[Rzepka, Sidoran, and White 1993] Rzepka, W.E., Sidoran, J.L., and White, D.A., "Requirements Engineering Technologies at Rome Laboratory," The First International Symposium on Requirements Engineering, 1993, pp. 15-18.

[Schneider, Easterbrook, Callahan, and Holzmann 1998] Schneider, F., Easterbrook, S. M., Callahan, J. R., and Holzmann, G. J., "Validating Requirements for Fault Tolerant Systems using Model Checking," The Third International Conference on Requirements Engineering, April 1998, pp. 4-13.

[Sommerville 1998] Sommerville, Requirements Engineering – A Practice Guide, John Wiley and Sons, 1998.

[Sutcliffe 2001] Sutcliffe, A.G. "Requirements Engineering for Socio-Technical Systems," in Proceedings Fifth IEEE International Symposium on Requirements Engineering, Toronto, 27-31 August 2001, Los Alamitos CA: IEEE Computer Society Press.

[Thayer and Dorfman 1990] Thayer, R. H. and Dorfman, M., Systems and Software Requirements Engineering, IEEE Computer Society, 1990.

[Yau and Tsai 1986] S. Yau and J. Tsai, "A Survey of Software Design Techniques," IEEE Transactions on Software Engineering, Vol. SE-12, No. 6, pp. 713-721, June 1986.

[Yeh 1982] Yeh, R. T., "Requirements Analysis – A Management Perspective," Proceedings of COMPSAC, pp. 410-416, 1982.

[Yourdon 1989] Yourdon, E., Modern Structured Analysis, Yourdon Press/ Prentice Hall, 1989.

[Yu 1993] Yu, E.S.K., "Modeling Organizations for Information Systems Requirements Engineering," The First International Symposium on Requirements Engineering, January 1993, pp. 34-41.

[Zave 1991] Zave, P. "An Insider's Evaluation of PAISLey," IEEE Transactions on Software Engineering, Vol. 17, No. 3. March 1991.

APPENDIX A. NON-FUNCTIONAL REQUIREMENTS AND CONSTRAINTS

A. Design Constraints
  - A.1 Design Constraints on Data
  - A.2 Design Constraints on Procedures

B. Performance Requirements
  - B.1 Timing
    - B.1.a Realtime
    - B.1.b Response Time
    - B.1.c Throughput
  - B.2 Space
    - B.2.a Computer Memory
    - B.2.b Available Data Storage Space
  - B.3 Productivity
  - B.4 Resource Utilization
  - B.5 Correctness
  - B.6 Comprehensiveness
  - B.7 Reliability
    - B.7.a Availability of Equipment
    - B.7.b Integrity of Information
  - B.8 Survivability
    - B.8.a On-Site and Off-Site Backup
  - B.9 Security
    - B.9.a Physical Security
    - B.9.b Operational Security
    - B.9.c Logical Issues
    - B.9.d Information Inference
  - B.10 Operating Constraints
    - B.10.a Frequency and Duration of Use
    - B.10.b Staffing
      - B.10.b.1 Availability of Personnel
      - B.10.b.2 Skill Level
    - B.10.c Hardware
    - B.10.d Software
    - B.10.e Control Procedures
    - B.10.f Remote and Local Monitoring
    - B.10.g Restart and Reconfigure
  - B.11 Economic Considerations
    - B.11.a Cost of Tradeoffs
    - B.11.b Cost of Iterative Systems Development
    - B.11.c Cost of Each Instance of Target Systems Delivery
    - B.11.d Immediate and Long-term Operations and Maintenance Costs
    - B.11.e Physical Constraints

- B.11 Physical Constraints
  - B.11.a Size, Weight, Power
  - B.11.b Portability
  - B.11.c Ruggedness
  - B.11.d Accessibility
  - B.11.e Space Distribution
  - B.11.f Maintenance
  - B.11.g Environmental Conditions
    - B.11.g.1 Temperature
    - B.11.g.2 Humidity
- B.12 Interface Constraints

C. Systems Development Requirements
- C.1 Kind of Development
  - C.1.a Traditional Systems Development
  - C.1.b Prototyping
- C.2 Scope and Scale of Effort
- C.3 Methodology
  - C.3.a Problem Definitions
  - C.3.b Systems Analysis
  - C.3.c Systems Design
  - C.3.d Programming, Conversion, Implementation
  - C.3.e Testing and Evaluation Factors
  - C.3.f Acceptance Criteria
- C.4 Quality Control Standards
  - C.4.a Methodological Standards
  - C.4.b Hardware
  - C.4.c Software
  - C.4.d Tool Usage
  - C.4.e Quality Assurance Programs
    - C.4.e.1 Closure
    - C.4.e.2 Completeness
    - C.4.e.3 Correctness
    - C.4.e.4 Consistency
    - C.4.e.5 Currency
    - C.4.e.6 Conciseness
    - C.4.e.7 Clarity
    - C.4.e.8 Cost-Effectiveness
    - C.4.e.9 Comprehensiveness
  - C.5 Priority and Changeability
  - C.6 Maintenability
    - C.6.a Enhanceability
    - C.6.b Portability

C.6.c Flexibility
C.6.d Reusability
C.6.e Compatibility
C.6.f Commonality
C.6.g Generality
C.6.h Modularity
C.6.i Independence

D. Implementation Constraints
- D.1 Programming Rules
- D.2 Testing Methods
- D.3 Training Programs
- D.4 Conversion Procedures
- D.5 Documentation Standards

E. Managerial Considerations
- E.1 Policy and Legal Issues
- E.2 Organizational Factors
- E.3 Management Concerns and Support
- E.4 Behavioral Considerations